Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for loading applications into modules of an embedded system having a runtime environment including a virtual machine comprising an intermediate pseudocode language interpreter and application programming interfaces (API), from a station on which a source code of the application is written, compiled into pseudocode by a compiler ~~(82)~~, verified by a verifier ~~(91)~~, converted by a converter ~~(92)~~ and loaded by a loader ~~(93, 68), is characterized in that~~ comprising:
[[-]] assigning an identifier to each module ~~(MID)~~, and a reference number to each class ~~(NCI)~~, each method ~~(NM)~~ and each attribute ~~(NA)~~ encapsulated in classes of a module, so as to statically link a plurality of sets of packages to be stored in the same name space in the embedded system to effect conversion by the converter ~~(92)~~ ;
[[-]] coding the reference to a method or an attribute in the linked pseudocode of a module in three bytes constituted by an indicator indicating the reference to a class internal ~~(II)~~ or external ~~(IE)~~ to the module, the number of the class ~~(NCI)~~, and either the number of the method ~~(NM)~~ or the number of the attribute ~~(NA)~~, and
[[-]] loading one or more application program interface modules comprising system classes or service modules, each corresponding to an application, a reference ~~(IE)~~ to an external class being systematically interpreted by the virtual machine as a reference to an application program interface module.

2. (Currently Amended) A method for loading applications into ~~an~~ the embedded system according to claim 1, ~~characterized in that~~ wherein the loading of the modules into the embedded system comprises storing of at least one module array ~~(69, 101, 103)~~ representing the modules, the number ~~(IMi)~~ between 0 and n associated by the linker with a module constituting the index of said module in the array, storing the association of the index in the array representing the identifier ~~(MID)~~ of said module in a table ~~(70, 102,~~

~~104)~~, said array and the table being in programmable nonvolatile memory, an external reference ~~(IE)~~ to an external module in the pseudocode ~~interpreting~~ interpreted as constituting an index for access to the module equivalent to the one in the module array.

3. (Currently Amended) A method for loading applications into ~~an~~ the embedded system according to claim 2, ~~characterized in that~~ wherein the loading of the modules comprises the storage, for each module, of an array ~~(TRC)~~ representing its classes, comprising a reference to the index of its module and, for each class, an array representing attributes ~~(TRA)~~ and methods ~~(TRMe)~~.

4. (Currently Amended) A method for loading applications into ~~an~~ the embedded system according to claim 2, ~~characterized in that~~ wherein the modules are referenced in a single module array, ~~interpreting~~ and the system classes different from n will be interpreted by the virtual machine as a reference to said API module.

5. (Currently Amended) A method for loading applications into ~~an~~ the embedded system according to claim 4, ~~characterized in that~~ wherein the classes ~~being~~ are declared public, or in private packages, the attributes and methods ~~being~~ are declared protected, in private packages or in private classes, the numbering of the classes is done in order of the public classes followed by the classes in private packages; the numbering of the attributes or methods is done by the converter ~~(92)~~ in order of the attributes or methods that are public, protected in private packages, and in private classes.

6. (Currently Amended) A method for loading applications into ~~an~~ the embedded system according to claim 2, further comprising containing system classes contained in several API modules that can be loaded separately, maintaining in the programmable nonvolatile memory two arrays representing modules and two corresponding MID/IMi association tables, one for the API modules and the other for the non-API modules, and loading the modules into one of the two arrays based on the nature of the module specified in ~~its~~ a header, any external reference of a module of the module array being interpreted as a reference to the index of the API module.

7. (Currently Amended) A method for loading applications into ~~an~~ theembedded system according to claim 6, ~~characterized in that~~ wherein static linking of a module is performed such that the reference to a class external to a non-API module in the intermediate pseudocode is an index in an array of the header of the module, wherein each entry is an identifier (MID) of a referenced API module, the loading of said module into the target platform comprising the replacement of said reference with the number of the index of the API module obtained from the identifier (MID) in the MID/IMi association tables of the API modules.

8. (Currently Amended) An embedded system comprising a virtual machine and an API platform including application program interfaces, a fixed nonvolatile memory, a programmable nonvolatile memory, and a random access memory, the programmable nonvolatile memory comprising at least one API module comprising system classes and service modules, at least one array representing modules ~~(TRM)~~, in which the modules are indexed, and a table ~~(70, 104)~~ associating an index ~~(IM)~~ of a module in the representing array with an identifier ~~(MID)~~ of said module, each module comprising an array for representing classes ~~(TRC)~~, in which the classes are indexed and in which each class has a reference to the index ~~(IM)~~ of its module, each class comprising an array for representing attributes ~~(TRA)~~ and methods (TRMe), in which the attributes and methods are indexed, the reference to a method or an attribute being coded in at least three bytes corresponding to a reference to a class internal ~~(II)~~ or external ~~(IE)~~ to the module, a reference external to the module constituting the index of the API module in the module array, a class number ~~(NCI)~~ corresponding to the index of the class in the table representing the classes of the module, and a method ~~(NM)~~ or attribute ~~(NA)~~ number corresponding to the index or the method or the attribute in the array representing the methods or attributes of the class of the module.

9. (Currently Amended) ~~An~~ The embedded system according to claim 8, further comprising means for comparing the first byte of the three bytes ~~encoding a reference to a method or an attribute~~ to a given value n in order to decide whether it is an internal or an external class.

10. (Currently Amended) ~~An~~ The embedded system according to claim 8, wherein a main module comprises the main program of the system.

11. (Currently Amended) ~~An~~ The embedded system according to claim 8, ~~characterized in that it~~ wherein the classes are indexed in order of the public classes followed by the classes in private packages, and the attributes and methods are indexed in the order of the attributes or methods that are public, protected, in private packages, and in private classes.

12. (Currently Amended) ~~An~~ The embedded system according to claim 11, characterized in that the programmable nonvolatile memory comprises several API modules comprising system classes, two arrays ~~(101, 103)~~ representing modules, one for the API modules and the other for the non-API modules and the main module, and two MID/IMi association tables ~~(102, 104)~~, each corresponding to an array representing modules.

13. (Currently Amended) ~~An~~The embedded system according to claim 10, further comprising an access manager class "Access Manager" of an API module ~~(105)~~ comprising a method for creating an instance of a service module, via the main module, said class having a protection that prohibits it from having more than one instance.

14. (Currently Amended) A method for executing an application of a multi-application embedded system having a runtime environment including a virtual machine comprising an intermediate pseudocode language interpreter and application programming interfaces (API), ~~is characterized in that~~ wherein during the execution of the intermediate pseudocode of a service module, corresponding to an application, referenced in a module array, the reference to a method or an attribute in the pseudocode, coded in at least three bytes corresponding to a reference to a class internal ~~(II)~~ or external ~~(IE)~~ to the module, a class number ~~(NCI)~~ and a method ~~(NM)~~ or attribute ~~(NA)~~ number, a reference external to the module is interpreted by the virtual machine as a reference to the index of an API module in the array of the API module or modules.

15. (Currently Amended) ~~A~~ The method for executing an application of a multi-application embedded system according to claim 14, c~~haracterized in that,~~ wherein -upon reception of a request for execution of a service module having an identifier ~~(MID)~~, the runtime environment accesses the input class of a main module comprising the main program of the system, the main module installs an instance of a special class "Access Manager" of an API module that controls access to a service module and uses a method of this class for creating an instance of the input class of the requested service module, by means of a table associating the identifier with the index of the module in an array in which the module is referenced, the instance being returned by the method to the main program.